

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования Национальный исследовательский Нижегородский
государственный университет им. Н. И. Лобачевского**

Институт информационных технологий, математики и механики

**Кафедра дифференциальных уравнений, математического и численного
анализа**

Направление подготовки

02.04.02. Фундаментальная информатика и информационные технологии

Направленность образовательной программы

**магистерская программа «Компьютерная графика и моделирование живых и
технических систем»**

Отчёт

**по методам глубокого обучения для решения задач компьютерного
зрения**

на тему:

**«Реализация метода обратного распространения ошибки для
двуслойной полностью связанной нейронной сети»**

Квалификация (степень)

магистр

Форма обучения

очная

Выполнил:

студент гр. 381706 – 2м
Бабаев Иван Владимирович

Нижний Новгород

2018

Оглавление

Цель.....	3
Задачи	3
Метод обратного распространения ошибки	4
Программная реализация.....	6
Приложение 1.	8

Цель

Цель настоящей работы состоит в том, чтобы изучить метод обратного распространения ошибки для обучения глубоких нейронных сетей на примере двухслойной полностью связанной сети (один скрытый слой).

Задачи

Выполнение практической работы предполагает решение следующих задач:

1. Изучение общей схемы метода обратного распространения ошибки.
2. Вывод математических формул для вычисления градиентов функции ошибки по параметрам нейронной сети и формул коррекции весов.
3. Проектирование и разработка программной реализации.
4. Тестирование разработанной программной реализации.
5. Подготовка отчета, содержащего минимальный объем информации по каждому этапу выполнения работы.

Метод обратного распространения ошибки

Метод обратного распространения ошибки (англ. backpropagation) — метод вычисления градиента, который используется при обновлении весов многослойного перцептрона.

Метод обратного распространения ошибки определяет стратегию выбора параметров сети с использованием градиентных методов оптимизации в предположении, что целевая функция $E(w)$ непрерывна. Градиентные методы на каждом шаге уточняют значения параметров, по которым проводится оптимизация, согласно формуле:

$$w(k + 1) = w(k) + \Delta w,$$

где $\Delta w = \eta p(w)$ определяет сдвиг значений параметров, η , $0 < \eta < 1$ – скорость обучения – параметр обучения, который определяет «скорость» движения в направлении минимального значения функции, $p(w)$ – направление в многомерном пространстве параметров нейронной сети. В классическом методе обратного распространения ошибки направление движения совпадает с направлением антиградиента.

Общая схема метода обратного распространения ошибки включает несколько основных этапов. Первоначально синаптические веса сети инициализируются определенным образом, например, нулевыми значениями или случайно из некоторого распределения. Далее метод работает для каждого примера обучающей выборки:

1. Прямой проход нейронной сети в направлении передачи информации от входного сигнала к скрытым слоям и выходному слою сети. На данном этапе вычисляются значения выходных сигналов нейронов скрытых слоев и выходного слоя, а также соответствующие значения производных функций активации на каждом слое сети.
2. Вычисление значения целевой функции и градиента этой функции.
3. Обратный проход нейронной сети в направлении от выходного слоя к входному слою, и корректировка синаптических весов.
4. Повторение этапов 1 – 3 до момента выполнения критериев остановки. В качестве критериев остановки используется число итераций метода (количество проходов), либо достигнутая точность.

Расчеты градиентов и обновления весов для двуслойной полносвязной нейронной сети с функцией активации *Softmax* на выходном слое и *Сигмоидальной* на скрытом слое представлены в Приложении 1.

Программная реализация

Была написана программа на языке C#, решающая задачу классификации рукописных цифр на основе базы данных рукописных цифр MNIST.

Класс NNet решает задачу обучения нейронной сети. Экземпляр класса создается со следующими параметрами:

- **inputImages** – список изображений тренировочной выборки.
- **hiddenLayerCount** – количество скрытых слоев.
- **outputLayerCount** – количество выходных слоев.
- **learningRate** – скорость обучения.
- **error** – заданная ошибка.

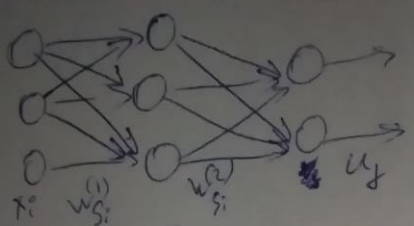
Класс реализует три публичные функции:

- **Train(int Epochs)** – Тренировать сеть на выборке изображений, указанное количество эпох.
- **Test(List<Image> image)** – Тестировать сеть на указанной выборке изображений.
- **AllocateInputData(Image image)** – выставляет параметры сети в соответствии с тестовым примером.
- **SigmoidActivation(double value)** – сигмоидальная функция активации.
- **SoftMaxActivation(double expSum)** – функция активации Softmax.
- **ComputeNeuronsHiddenValue()** – вычисление значений нейронов скрытого слоя.
- **ComputeNeuronsOutputValues()** – вычисление значений нейронов выходного слоя.
- **ComputeDerivatives(double[] factOutputSignals)** – вычисление градиентов выходного и скрытого слоев.
- **ChangeWeights()** – обновление весов.
- **ComputeCrossEntropy()** – подсчет кросс-энтропии.

Результаты работы программы:

Количество эпох	Количество нейронов скрытого слоя	Количество нейронов выходного слоя	Скорость обучения	Заданная ошибка	Точность
5	800	10	0.1	0.005	0,9628
10	800	10	0.1	0.005	0,9715
15	800	10	0.1	0.005	0,9761

Приложение 1.



$y^k = (y_j^k)_{j \in \overline{1, M}} \in Y$ - множество выходных функций,
 $u^k = (u_j^k)_{j \in \overline{1, M}}$ - выход сети, для входа $x^k = (x_i^k)_{i \in \overline{1, N}}$
 v_s - выходной сигнал нейрона скрытого слоя

В качестве функции активации на скрытом слое
 используем логистическую σ -f: $\sigma(x) = \frac{1}{1 + e^{-x}}$
 на выходном: $\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}$

В качестве функции ошибки - кросс-энтропии:

$$E(u) = - \sum_{j=1}^M y_j \ln u_j$$

Найдем производную по параметрам выходного слоя:

$$\frac{\partial E}{\partial u_{js}^{(2)}} = \frac{\partial (- \sum_{j=1}^M y_j \ln u_j)}{\partial u_{js}^{(2)}}, \quad u_j = \sigma^{(2)}(g_j) = \sigma^{(2)}\left(\sum_{s=0}^K w_{js}^{(2)} v_s\right)$$

$$g_j = \sum_{s=0}^K w_{js}^{(2)} \cdot v_s$$

$$v_s = \sigma^{(1)}(f_s) = \sigma^{(1)}\left(\sum_{i=0}^N w_{si}^{(1)} x_i\right)$$

$$f_s = \sum_{i=0}^N w_{si}^{(1)} \cdot x_i$$

$$u_j = \sigma^{(2)}(g_j) = \sigma^{(2)}\left(\sum_{s=0}^K w_{js}^{(2)} v_s\right) = \sigma^{(2)}\left(\sum_{s=0}^K w_{js}^{(2)} \sigma^{(1)}(f_s)\right) =$$

$$= \sigma^{(2)}\left(\sum_{s=0}^K w_{js}^{(2)} \sigma^{(1)}\left(\sum_{i=0}^N w_{si}^{(1)} x_i\right)\right)$$

Подставим:

$$\frac{\partial E}{\partial u_{js}^{(2)}} = \frac{\partial (- \sum_{j=1}^M y_j \ln \sigma^{(2)}(\sum_{s=0}^K w_{js}^{(2)} \sigma^{(1)}(\sum_{i=0}^N w_{si}^{(1)} x_i)))}{\partial u_{js}^{(2)}} =$$

$$= - \frac{y_j}{u_j} \frac{d\psi^{(2)}(g_j)}{dg_j} \frac{dg_j}{dw_{js}^{(2)}} = - \frac{y_j}{u_j} \frac{d\psi^{(2)}(g_j)}{dg_j} v_s = \delta_j^{(2)} v_s$$

$$\delta_j^{(2)} = - \frac{y_j}{u_j} \frac{d\psi^{(2)}(g_j)}{dg_j} = \frac{\partial E(u)}{\partial g_j}$$

$$\hookrightarrow \frac{\partial E}{\partial w_{js}^{(2)}} = \delta_j^{(2)} v_s$$

Найдем производную по параметрам скрытого слоя

$$\frac{\partial E}{\partial w_{s_i}^{(1)}} = \frac{\partial \left(- \sum_{j=1}^M y_j \ln \psi^{(2)} \left(\sum_{s=0}^K w_{js}^{(2)} \cdot \psi^{(1)} \left(\sum_{i=0}^N w_{s_i}^{(1)} x_i \right) \right) \right)}{\partial w_{s_i}^{(1)}} =$$

$$= - \sum_{j=1}^M \frac{y_j}{u_j} \frac{d\psi^{(2)}(g_j)}{dg_j} \frac{dg_j(v_s)}{dv_s} \frac{d\psi^{(1)}(f_s)}{df_s} \frac{df_s}{dw_{s_i}^{(1)}} =$$

$$= - \sum_{j=1}^M \frac{y_j}{u_j} \frac{d\psi^{(2)}(g_j)}{dg_j} w_{js}^{(2)} \frac{d\psi^{(1)}(f_s)}{df_s} x_i = \delta_s^{(1)} x_i$$

$$\delta_s^{(1)} = - \sum_{j=1}^M \frac{y_j}{u_j} \frac{d\psi^{(2)}(g_j)}{dg_j} w_{js}^{(2)} \frac{d\psi^{(1)}(f_s)}{df_s} = \frac{\partial E(u)}{\partial f_s}$$

$$\hookrightarrow \frac{\partial E}{\partial w_{s_i}^{(1)}} = \delta_s^{(1)} x_i$$

Производная по параметрам
линейного слоя:

$$\frac{\partial E}{\partial w_{js}^{(1)}} = - \frac{y_j}{u_j} \frac{e^{g_j} \sum_{k=1}^M e^{g_k} - e^{g_j}}{\left(\sum_{k=1}^M e^{g_k} \right)^2} v_s = \delta_j^{(2)} v_s$$

Производная по суммарной:

$$\frac{\partial E}{\partial w_{s_i}^{(1)}} = - \sum_{j=1}^M \frac{y_j}{u_j} \frac{e^{g_j} \sum_{k=1}^M e^{g_k} - e^{g_j}}{\left(\sum_{k=1}^M e^{g_k} \right)^2} =$$

$$= \frac{1}{1+e^{-f_s}} \left(1 - \frac{1}{1+e^{-f_s}} \right) w_{js}^{(2)} x_i = \delta_s^{(1)} x_i$$